VLSI Architectures for Digital Signal Processing on Energy Constrained Systems-on-Chip

Alicia Klinefelter
Department of Electrical and Computer Engineering
University of Virginia

A Dissertation Proposal Presented in Partial Fulfillment of the Requirement for the Doctor of Philosophy Degree in Electrical Engineering

April 23, 2013

Abstract

The design of ultra-low power integrated circuits for system-on-chip (SoC) architectures requires both robust and reliable low-energy operation. Promising applications for these SoCs include body sensor nodes (BSNs) capable of running on energy harvested from the surrounding environment alone. The strongest design knob for reducing energy on chip is to lower the supply voltage and operate digital circuits in the subthreshold region, which is using a supply that falls below the threshold voltage of the device. Although operating in subthreshold also leads to a large performance penalty, there are still many applications with low throughput requirements that can take advantage of subthreshold operation's energy savings. Systems that process biomedical data such as EKG, EEG, and EMG require sampling rates less than 500Hz, making subthreshold operation attractive in this application space.

The SoCs presented in this work often involve high-levels of integration between many blocks that are in the digital, analog, and mixed-signal domains. Due to this complexity, evaluating chip and bus architectures to optimize a given metric (e.g. speed, power) can be time-intensive using traditional circuit design and simulation tools such as Cadence and SPICE. Similarly, component-level designers need to make lower-level architecture decisions and be able to quickly compare designs and evaluate their interactions with other chip components. The development of a modeling tool that captures the behavior, energy, performance, and area of typical blocks on a low-power SoC and automates the design decision process is extremely valuable.

This work introduces and explores architecture techniques for traditional digital signal processing (DSP) algorithms (e.g. linear filters) that operate in the subthreshold region on a biomedical SoC. Techniques such as resource sharing of circuits for serialized operation and the approximation of filtering coefficients on chip are evaluated. Next, a component and chip-level modeling tool, created in Simulink, is introduced for evaluating the power-area-performance design space for a given block or chip architecture to justify design decisions. For model validation, various DSP designs are proposed including filters, a Fast-Fourier Transform (FFT), bus controllers, and an all-digital phase-locked-loop (ADPLL) that has complex control, filtering, multi-rate functionality, and feedback that captures the behavior of a wide-range of possible system blocks.

1. Introduction

1.1. Motivation for Energy Efficient on-chip DSP

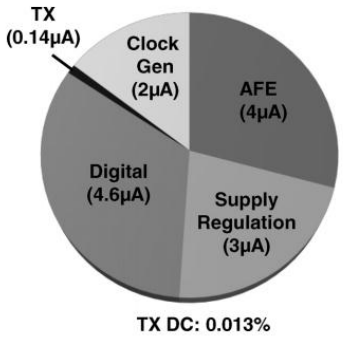
Continuous health monitoring for early detection of chronic diseases or fitness tracking is an emerging trend due to smaller product form factors as a result of technology scaling. With an increase in life-expectancy, a large aging population, and a desire for autonomy amongst the elderly, research in the area of body sensor nodes is increasing to provide devices for unobtrusive and precise monitoring [1]. To increase the adaptability of BSNs, they should be non-invasive, reliable, safe, secure, and have long lifetimes to avoid continuous battery replacement [3]. Due to these criteria, BSNs and other ultra-low-power (ULP) systems demand energy efficient signal processing to meet their stringent power requirements. For example, a BSN SoC with four input channels for electrocardiography (EKG)/ electroencephalography (EEG)/electromyography (EMG) (ExG) data, analog-to-digital converter (ADC), microcontroller (MCU), radio, and power management runs entirely on power harvested from body heat with no battery [2]. To enable this, the chip must consume an average power less than the power harvested, which is typically in the 30-50µW range. When extracting heart rate from EKG and sending regular RF updates, the entire chip consumes just 19µW [2]. Since the chip harvests power, the processing blocks are power limited to keep the total average chip power below the harvested power and avoid depleting energy on the storage capacitor.

On the first revision of the BSN, an open-source PIC microcontroller (MCU) was used for general purpose processing. Relative to custom accelerator blocks on chip, the PIC was found to be energy inefficient when completing the same task as the hardware accelerators as seen in Figure 1a. MCUs are highly flexible but are not energy or area efficient. They work well for sequential instructions and operations, but do not allow for a tradeoff between serial and parallel architectures to adapt to varying application throughputs [4]. On this chip, four subthreshold accelerators reduced computation delay and increased energy efficiency compared to the MCU for frequently-used functions such as filtering, signal power determination, and R-R interval extraction for EKG data.

Looking at the current breakdown of the BSN chip shown in Figure 1b, a third of the chip power is consumed by the digital circuitry. Since this is a significant slice of the total current consumed by the chip, energy optimizations in the digital domain can make a large impact to the overall chip power. Since the application space for BSNs spans data transfer rates from 1-10,000bps with lifetime requirements from seconds (RFID) to years (temperature, light, pressure), flexible designs with high levels of programmability are needed [3]. The presented BSN node aims to be flexible and low power while meeting its throughput requirements for a variety of applications from ExG to speech processing. For a hardware accelerator, increased programmability leads to a power-flexibility tradeoff that is evaluated in this work.

Energy Efficiency Comparison Per Sample		
30 Tap FIR	MCU	6.3 nJ
	Accel	1.9 pJ
Signal Energy Extractor	MCU	3.6 nJ
	Accel	530 fJ
R-R Interval Extractor	MCU	12 pJ
	Accel	3 fJ

(a)



(b)

Figure 1: (a) Processing energy using the MCU versus an accelerator circuit (b) Current breakdown for BSN chip in [2].

1.2. Prior Art

1.2.1. Subthreshold Operation

Subthreshold conduction refers to lowering the supply voltage below a device's threshold voltage, V_t, so that the device operates using leakage currents. Active energy scales quadratically with reductions in the supply voltage, making it a useful knob for designs where energy efficiency is the driving metric and performance is secondary. Besides the performance degradation that comes with subthreshold operation, there are also nonidealities such as increased sensitivity to variation due to an exponential dependence of the drain current of a device on V_t. The threshold voltage can vary on chip due to non-uniform doping or lithography, and the impact on performance variability is pronounced in this region.

Most digital design is automated using hardware description language (HDL) and EDA tools, and there are various techniques used for subthreshold synthesis such as avoiding ratioed logic (SRAM bitcells) and large device stacks (NOR gates) [5]. Standard cells with these characteristics are avoided since they degrade the reliability of the design to the point that the circuit may not work reliably at low voltages.

1.2.2. Flexible and ULP BSN Design

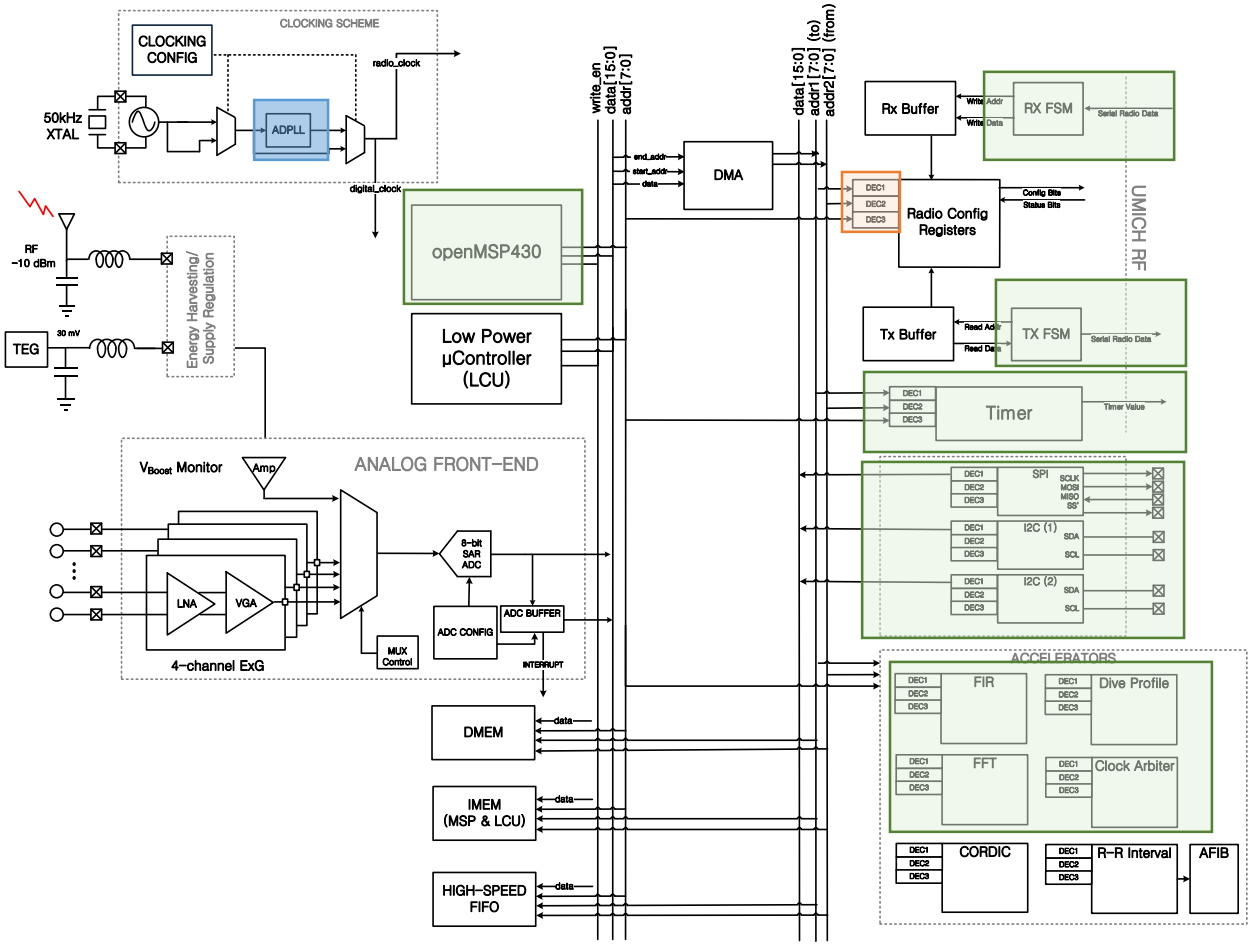


Figure 2: Block-level BSN chip architecture taped-out in February 2013. Blocks discussed in this document have been highlighted.

The integrated system diagram for a recently fabricated BSN SoC is shown in Figure 2. The node contains an analog front-end (AFE) for amplifying and digitizing data from analog sensors, a transceiver for low-energy communication, a low-power clocking scheme using a 50kHz crystal oscillator and ADPLL, and a subthreshold digital section including power management, a low-power microcontroller (LCU), an open source MSP430 [6] for flexible computation, direct memory access (DMA), accelerators, six SRAMs (data memory, program memory, and TX/RX buffers), and serial interfacing circuits (I2C/SPI). The chip has a memory mapped architecture that relies on block decoders to control the flow of data on the bus. The block decoder design and address range assignment was automated using scripts to reduce the chance of errors from manual decoder modifications. The first bus can be controlled by the LCU or an MSP430 microcontroller. For modes where power is critical, the LCU controls the source and destination locations of data on the bus. Since this chip was completed with system developers in mind, the MSP430 was also used as a bus controller due to an available toolchain that can be used to compile C code for the core. The second bus is controlled using a DMA unit that can efficiently stream multiple words of data from one location to another with only a few setup instructions. My contributions to this chip included:

1. Low-power decoder design with Perl script backend for automated Verilog code generation for arbitrary block decoders
2. A timer for general purpose capture and compare as well as timestamp generation for packet transmission using an ultra-wide band transmitter
3. Finite state machines (FSMs) to interface between the transmitter(TX)/receiver(RX) and the digital portion of the chip
    a. The TX FSM serializes data from the TX buffer, appends preamble bits to the front of the packet, and a timestamp to the end of the packet
    b. The RX FSM checks a preamble to determine the packet destination, and then parallelizes data to be stored in the RX buffer
4. I2C/ SPI protocol wrapper, and high-speed I2C burst block (these blocks operated at the nominal voltage for this technology, not in subthreshold)
    a. I2C/SPI logic was provided by OpenCores, but used a Wishbone Interface. A decoder wrapper and FSM was added to these blocks for easy data transmission/receives using minimal bus instructions. The high-speed memory use in the high-speed burst block was designed by Jim Boley.
5. OpenMSP430 HDL modifications for bus arbitration between the LCU and MSP430
6. Accelerators
    a. Finite-impulse response (FIR) filter
    b. 64- point, radix-2 Fast-Fourier Transform (FFT)
    c. Clock arbiter for dividing down the system clock for blocks with lower frequency requirements
    d. Dive profile block used for detecting whale dives from depth sensor measurements (application not discussed here)

The system architecture and composite blocks of this chip are a motivator for the design methods discussed in this document, and the proposed designs and ideas work primarily within this scope. The focus of the architecture optimization discussed in section (2) of this document pertains mostly to classical DSP structures such as filters and FFTs. The system modeling effort in section (3) discusses the design of a framework for the exploration of low power architectures for arbitrary digital blocks. Section (4) explores the design and modeling of an ADPLL used for on-chip clock synthesis and model validation.

1.3. Research Goals

The following is a list of research goals and contributions for the presented work, with each bullet discussed in more detail in the following sections:

1. Design, fabricate, and test a low power, flexible state of the art BSN node that uses the subthreshold digital blocks described in (1.2.2).
2. Investigate VLSI architectures for FIR filters, IIR filters, and Fast-Fourier Transforms and evaluate the power, throughput, and area tradeoffs.
3. Use on-chip FIR filter coefficient generation for low-leakage filtering of high-tap filters.
4. Complete a custom modeling tool for SoC design space exploration and design automation.
5. Design the first subthreshold all-digital phase-locked loop (ADPLL) for model (4) validation and low-power on-chip clocking.

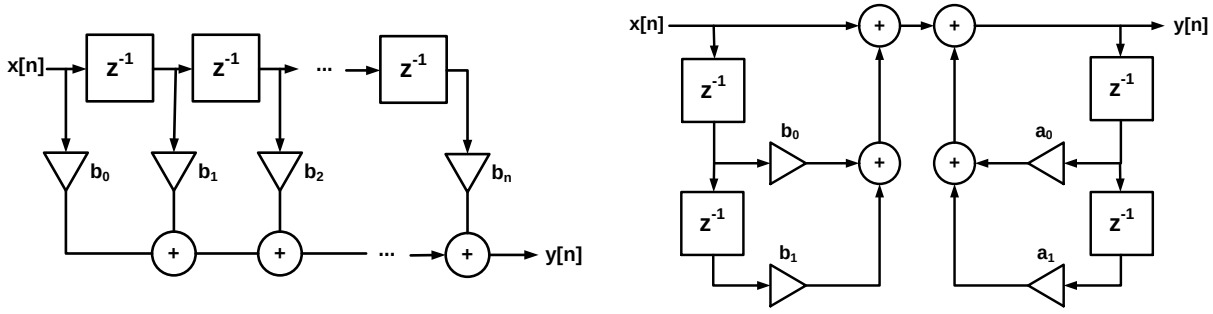
1.4. Thesis Statement

Energy efficiency is the main focus for biomedical SoC design, with supply voltage scaling down to the subthreshold region being the means to achieve it. For digital circuit design, this means using novel VLSI architectures that minimize energy consumption at the expense of degraded performance. The impact of these design choices is evaluated in the context of ULP system design, and custom tools for ASICs are developed for fast and accurate design decision comparisons.

2. DSP Block Design for Energy Constrained Systems

2.1. Overview and Motivation

DSP algorithms are generally divided into two categories: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). The latter is often defined based on the output's dependence on past and present inputs (i.e. recursive), while FIR is dependent only on the present input as seen in Figure 3. To compute each output, y[n], the algorithm must complete one iteration where the delay between the system receiving the input and producing a corresponding output is the system latency [7]. These systems require one multiplication and addition operation per tap, and the number of taps used determines the accuracy of the system. FIR filtering is most commonly seen in the literature due to a guaranteed system stability and linear phase characteristic. IIR filtering has the advantage of needing fewer computing resources to achieve the same performance as an FIR, but can become unstable due the feedback structure. IIRs also have a nonlinear phase characteristic that is not ideal for certain applications such as speech/audio, although methods to linearize the phase have been presented in the literature as in [8].



(a) *(b)*

Figure 3: (a) Direct-form FIR filter topology (b) Direct-form IIR filter topology

Filtering is frequently required for processing EEG data for band-specific spectral power analysis and is used for detecting various neurological states such as sleep or motor movement shown in [9][10][11][12]. IIR filtering topologies were considered for the BSN, but were not used due small filter bandwidths for sleep applications (e.g. 0.5-3Hz with a sampling rate of 256Hz), putting the design at a high risk for errors due to unstable operation not accounting for possible quantization error.

Fast-Fourier Transforms (FFTs) are another commonly used class of DSP that transform time-domain data to the frequency domain. BSN target applications such as sleep phase detection, speech and wheezing detection algorithms use FFTs in the processing loop. An FIR filter and FFT were designed and fabricated for a BSN SoC and are presented here.

2.2. Hypothesis

Directly mapping traditional DSP algorithms to hardware leads to suboptimal designs if energy is the driving metric. The high-level of parallelization of adders and multipliers leads to increased area and leakage. This area penalty is worsened when the filter or transform needs multiple processing channels to process independent streams of data from a multi-channel AFE. Serializing DSP architectures can drastically reduce area and energy while still meeting the throughput demands for a variety of BSN applications. Furthermore, the designed BSN node is targeting a large, varied set of applications, which requires flexibility/programmability of the digital processing blocks.

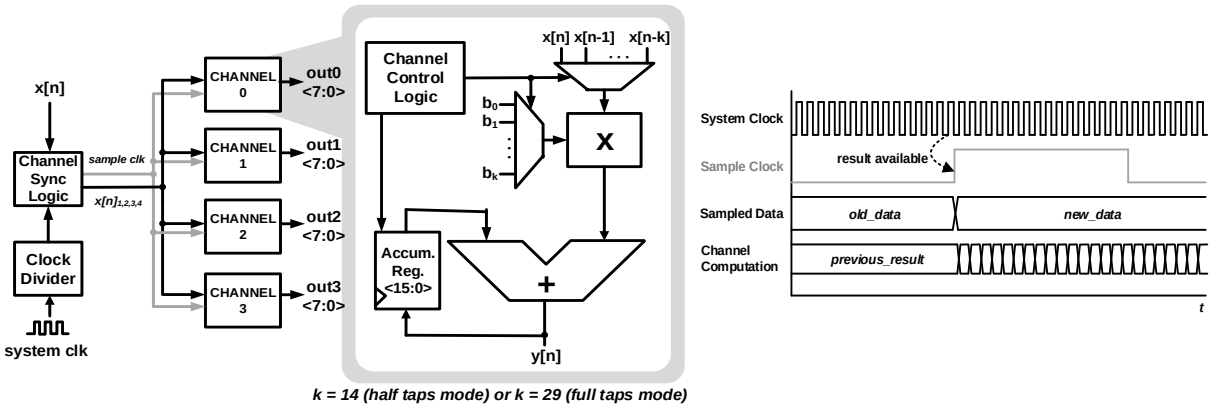
2.3. Approach

2.3.1. Finite Impulse Response (FIR) Filter Design

To support concurrent ExG processing on data from the four analog input channels on chip, the architecture of the FIR has four independent channels as seen in Figure 4a. The SoC uses a 200kHz XTAL for the system clock to modulate data in the Medical Implant Communication Service (MICS) band radio and for the digital clock. Since the typical data sampling rate for ExG is much lower than 200kHz (<1024 Hz for our chip), we use a serial architecture [4] versus the traditional direct-form architecture for a FIR filter to reduce power, area, and leakage. The direct-form FIR architecture computes the result in parallel using as many adders and multipliers as there are taps as seen in Figure 3a. This method results in a much higher throughput than required for this application space as well as a large area penalty for up to 30 taps replicated over four channels. Resource reuse of the arithmetic units was achieved using the faster 200kHz clock for serially processing data between receiving input samples. This allowed for a higher utilization of arithmetic units per sample, resulting in lower leakage from inactive or excess circuitry when the design is not on the datapath. At every rising edge of the sampling clock, a new input sample is received, processed at the faster clock rate, and the result is computed in a fraction of the sample period as seen in Figure 4b.

Each channel contains one 8-bit Baugh-Wooley multiplier, one 16-bit ripple-carry adder, coefficient registers, and a small filter controller for channel synchronization and state retention of the channels. During active operation, each channel is individually clock-gated for any remaining cycles after the result is computed to further reduce energy. Sharing arithmetic units within each channel reduced the overall area by 6-12X/filter compared to prior work [13][14] and by 12X/channel compared to a traditional parallel architecture. The smaller area also reduces leakage, which helps decrease energy drawn from the off-chip storage capacitor during idle periods.

The filter supports programmability by the SoC's MCU for several modes of operation. Due to the volatility of energy on the storage capacitor during power harvesting, several programmable modes were included in this design to reduce energy consumption. When high throughput and energy are more critical than accuracy, a 15-tap mode is available. Each stream of input data can be filtered using one channel or two simultaneously with different filtering coefficients.



(a) *(b)*

Figure 4: (a) Resource-shared architecture of the 4-channel FIR including detailed diagram of the channel. (b) Timing diagram describing filter operation

The filter was synthesized using only standard cells and fabricated in 130nm CMOS as part of the BSN SoC. It operates correctly across the target range of 0.3V-0.7V with a corresponding frequency range of 8kHz-6MHz. Clock gating channels after the result is ready reduces switching energy by 4X, and the MCU power gates the filter using PMOS headers when the block is unused, reducing leakage by up to 15X. Recent sub-threshold FIR filters have included fewer than 15 taps, consumed more power, more area, and have a much larger FOM without having the flexibility demonstrated in this design as seen in Table I. To date, this filter has the smallest FOM compared to the state of the art.

	This Work	[13]	[14]	[10]
Type	30-tap, 8-bit	8-tap, 8-bit	14-tap, 8-bit	4th order analog
Channels	4	1	1	4
Programmable	✔	✘	✘	✔
Technology	0.13µm	0.13µm	0.13µm	0.13µm
Supply	350mV	200mV	270mV	1.2V
Frequency	29kHz	12kHz	20MHz	20kHz
Energy/Tap	1.10pJ	1.19pJ	1.11pJ	(total) 39pJ
Power	32nW	114nW	310µW	780nW
FOM*	0.57	18.55	17.37	N/A
Area/Channel	0.058mm^2	1.54mm^2	0.38mm^2	0.7mm^2

*Table 1: FIR comparison table. *FIR FOM: power(nW)/frequency(MHz)/# of taps/input bit length/coefficient bit length.*

2.3.2. Fast-Fourier Transform (FFT) for Flexible BSN Systems

A complex-valued, 64-point FFT circuit was fabricated on the latest BSN chip and uses an in-place addressing scheme for area savings similar to that in [17]. 16-bit Baugh-Wooley multipliers were used in the radix-2 butterfly unit for low-power operation, and one butterfly was computed per system clock cycle. The applications that require the FFT for frequency domain analysis have a wide range of sampling rates from 200Hz(EKG)-2kHz (speech). A 64-point hardware FFT takes hundreds of clock cycles to compute the result, so that large buffers to store incoming samples are needed for applications with high Nyquist rates. Lower rate applications such as ExG can still process the data in real-time. Increasing the radix of the FFT increases the throughput, but doubles the amount of processing resources required

per clock cycle. A radix-4 FFT butterfly is shown in Figure 5, which is comprised of two radix-2 butterflies. A butterfly circuit combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT using multiplication and addition. An all radix-2 FFT can be used for lower throughput applications to reduce energy per operation, while the radix-4 butterfly can be used in some FFT stages for a performance improvement. This introduced flexibility will increase the amount of control logic for addressing and butterfly input data selection, but will accommodate the various application requirements targeted for BSNs. These tradeoffs will be explored using the modeling tool discussed in section (3).

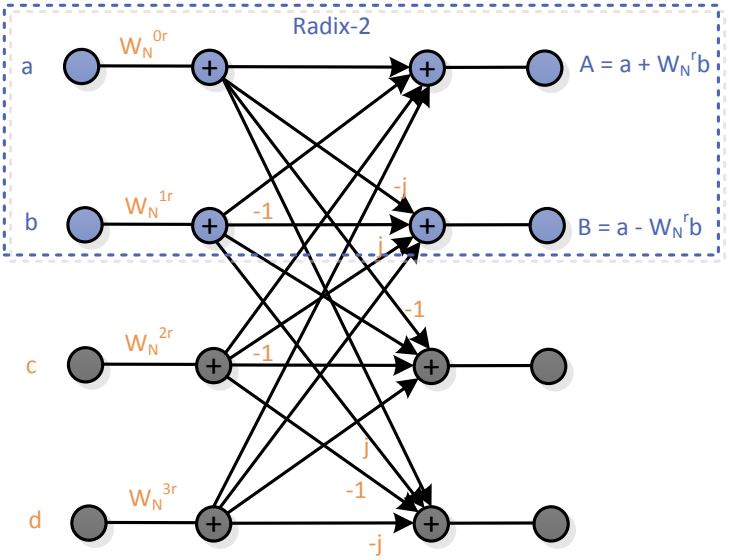


Figure 5: Butterfly structure for a radix-4 FFT. The radix-2 FFT is a subset of the radix-4 and is highlighted in the figure.

2.4. Contributions and Anticipated Challenges

Contributions and metrics for success (sub-bullets) include:

1. A fabricated and tested multi-channel subthreshold FIR filter with the best FOM in its class
    a. Operational FIR circuit tested in silicon. Compare energy, area, performance, and FOM to the state of the art
2. Next revision of FIR filter with flexible number of taps (not just 15 and 30) and channel-to-channel "daisy-chaining" to increase the filter order. This was designed and fabricated for the BSN node taped out in February 2013.
    a. Operational FIR circuit tested in silicon. State of the art comparison (same as 1a).
3. A FFT circuit that can be programmed for radix-2, radix-4, or mixed operation for an energy-throughput tradeoff
    a. Operational mixed-radix FFT circuit tested in silicon
    b. Analysis of the energy-throughput tradeoffs due to the mixed-radix design. Determine an estimate for the area and energy overhead due to the added butterfly and control logic using the modeling tool discussed in (3).

Anticipated challenges:

1. The critical path in the FFT is contained in the butterfly through a series of adders and multipliers. Maintaining robust operation in subthreshold at frequencies >200kHz will be a

design challenge. The butterfly could include an optional pipeline stage for higher frequency operation, or the clock arbiter could include a dedicated FFT output in future revisions.

2.5. Dynamic Coefficient Approximation for Filters

2.5.1. Overview and Motivation

For a BSN node, an FIR filter's coefficients are often stored in an on-chip SRAM or register file that is also operating in subthreshold. For filters that require a high number (> 15) of taps, this leads to large portions of the data memory being consumed by coefficient data, and additional bus congestion required to move coefficients from the data memory to the filter during operation. The chip's data memory is rarely power-gated as it also stores chip data that may be unrelated to the filtering process. This leads to excess leakage due to stored coefficients when the filter is not being used. On-chip SRAMs often consume ~50% of a chip's area and if not power gated can dominate the digital power budget for a SoC. Adding to the issues associated with SRAMs in subthreshold, decreased I_{on}/I_{off} ratios reduce the reliability of their operation deep in the subthreshold region [18]. A 1.5kB 8T SRAM was designed by the RLP-VLSI group at UVA, and measured results show reliable operation down to 0.3V at 200kHz with read energy of 12.1pJ at 0.5V (0.378fJ for 16-bits) and leakage energy per cycle of 6.6pJ at 200kHz (0.206fJ for 16-bits) . For a 16-bit word, it was observed that it would consume 275.2um^2 of area in a 130nm technology [2]. On the BSN chip that contained this SRAM design, the memory had two 0.75kB banks. Each bank can be separately power gated to save energy, but for most DSP operations, the amount of memory used is << 0.75kB and using a full memory bank is overkill. For another comparison point, a similar 1kb subthreshold SRAM was completed in a 65nm technology, and it consumed 115nW of active power and 105.9nW of leakage power at 300mV, 38kHz [18]. The next best solution is to use registers local to the accelerator to hold data and coefficients as in section (2), but this puts an upper bound on the order of the filter.

Another anticipated challenge for deployed BSNs is updating or reprogramming a node after deployment has occurred, and it's not ideal for end-users of BSNs to manually update the software on their devices. To fix this issue, the latest BSN node includes a program by wireless mode that can reprogram the instruction memory on-chip through the radios. Radio operation is very power hungry relative to other chip components, so it's best to rarely transmit/receive and limit the size of the transmitted payloads. If a new coefficient set needs to be sent for a new or adapted application, then the radio transmission would starve the chip of available power. If generated dynamically on chip, only a normalized cutoff frequency specification and filter order would need to be retransmitted.

2.6. Hypothesis

Dynamically generating filtering or transform coefficients on chip using approximation methods can lead to more robust, low power operation on BSN nodes. Filters requiring a large number of taps can benefit the most from this approach due to reduced reliance on unreliable subthreshold SRAMs for storage.

2.7. Approach

2.7.1. Theory

There are various known methods for generating filter coefficients such as frequency sampling, least squares, and windowing [7]. The first two are difficult to compute in an energy-constrained system due to a required FFT for the first and optimization to reduce the mean-squared error for the second. The windowing method is used in this work due to its relative computational simplicity. To generate coefficients using this method, consider the example of a lowpass filter described as a square pulse in the frequency domain shown in (1). An inverse discrete-time Fourier Transform (IDTFT) is taken to

obtain time-domain coefficients for filtering and this results in a sinc function, h[n], which is infinite, non-causal, and not realizable. Windowing functions are used to truncate the sinc to make h[n] finite. Choosing the right windowing function id dependent on the application as they all optimize for one of the following metrics: transition bandwidth, peak approximation error, and stopband attenuation.

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| \le \omega_0 \\ 0 & \omega > |\omega_0| \end{cases} \quad \overset{IDTFT}{\Longrightarrow} \quad h[n] = \frac{\sin(\omega_0 n)}{\pi n} \tag{1}$$

Generating h[n] requires the evaluation of a sinc function, and this requires an approximation for sine. The big tradeoff for this design is using an approximation method that isn't too complex to implement in hardware, but still has acceptable accuracy. To further reduce processing time and power, it's assumed that the produced coefficients will be symmetric so that the number of clock cycles required to produce the result for an N-tap filter is (N+1)/2, mathematically described in (2).

$$y(n) = \sum_{k=0}^{N/2} h(k)[x(n-k) + x(k)] \tag{2}$$

For a given input sample, this approach reduces the number of multiplications by a factor of two. Since trigonometric functions and division by non-powers-of-two are costly to realize in hardware, avoiding these methods is the best option to keep power and area low and still have this method be competitive against the SRAM storage solution.

The steps to realizing the windowing scheme in hardware are outlined below. Only lowpass filters are discussed here as they are the simplest and most straightforward to implement:

1. Define filter specifications such as the cutoff and stopband frequencies, windowing method, and filter order, *N*.
2. Generate an ideal impulse response function (sinc) for the given filter specifications. This is computed by creating an ideal frequency response, using the specifications defined in step (1), and then taking the IDTFT to obtain the coefficients. For this project, the DFT will not be implemented in hardware, but the final result will be used: $h(n) = \frac{\sin(\omega_c n)}{\pi n}$.
3. Delay the impulse response to keep the system causal:
   $$h(n) = \frac{\sin\left(\omega_c\left(n - \frac{N}{2}\right)\right)}{\pi\left(n - \frac{N}{2}\right)}$$
4. Multiply the sinc function by the specified window and sample to obtain coefficients (sample every 2π/N points) [19]. Many of the most common windows also contain cosine functions that need to be approximated. Shown below are three well-known window functions with varying computational complexities. Blackman is another commonly used window, but contains two cosine terms with differing arguments, and the added complexity works against the energy efficiency goal for this work:

Rectangular	$w[n] = 1$
Bartlett	$w[n] = 1 - \frac{\lvert n \rvert}{N+1}$
Hamming	$w[n] = 0.54 + 0.46\,cos\left(\frac{2\pi n}{2N+1}\right)$

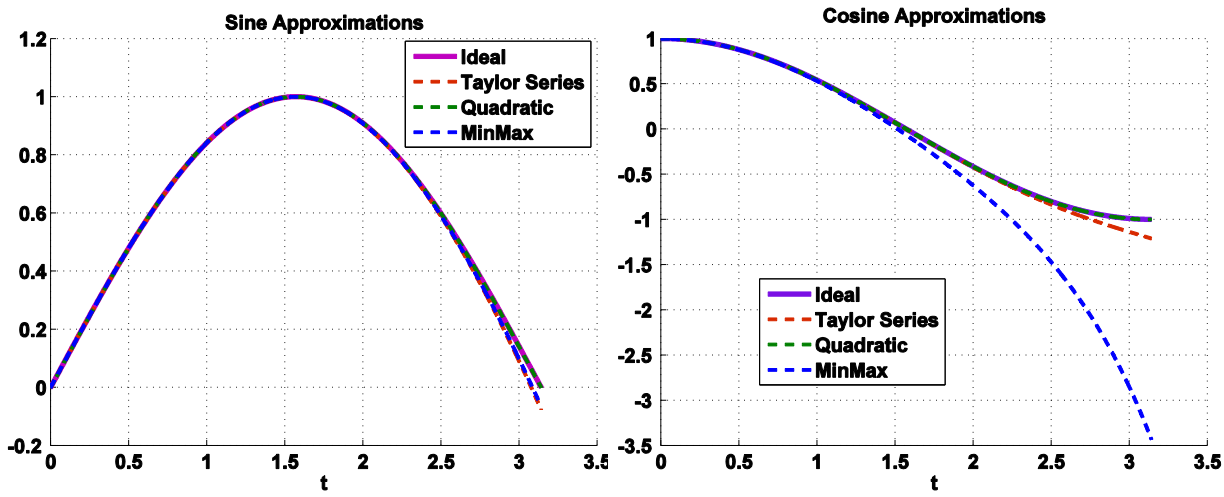
The majority of the discussion in this section will focus on on-chip sine and cosine approximations for steps (2-4) outlined aboveObtaining values for the impulse response involves multiple multiplication, division, and trigonometric approximation operations, none of which are trivial in hardware. Direct approximations of the sinc function have been developed, but have limited range (e.g. 0 to π/2) and are more computationally complex than the approximations discussed here. Three methods were considered and analyzed. A common approximation used for trigonometric functions is the Taylor series. For Taylor Series for the sine and cosine are shown in equations (3) and (4), respectively.

$$\sin\theta = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots = \theta - 0.16667\theta^3 + 0.00833\theta^5 - 0.0001984\theta^7 \ldots \quad (3)$$

$$\cos\theta = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + \cdots = 1 - 0.5\theta^2 + 0.041666\theta^4 - 0.0013888\theta^6 \ldots \quad (4)$$

Computationally, this is expensive to do for accurate results (i.e. more terms) and the error for this method is only acceptably low for small values of θ. We will assume that the approximations are used up to the 7th order for comparison, which required 9 multiplications and 4 additions to get the result for each iteration. Another method based on quadratic approximation of sine in the range of [0, π] was presented in [20].

The third method to be evaluated uses MinMax polynomials. This method is often used over the Taylor polynomial method due to its distributed error over the range [21]. All three of these methods were compared against the ideal sine and cosine functions in MATLAB and the approximations were plotted over the range of [0, π] as seen in Figure 6.



(a) *(b)*

Figure 6: Approximations for sine and cosine using the Taylor Series, Quadratic, and MinMax methods.

For both functions, the quadratic approximation is the most accurate. Since we will make the assumption that the following approximations are being implemented in a 16-bit, fixed-point system, we then compared the quantized version of the quadratic approximation to the ideal. 8 bits were used for the integer and decimal representation to mimic a real 16-bit system. Using this approximation, sets of coefficients for various number of taps were determined for a lowpass filter with a normalized cutoff frequency of 0.5π.The frequency responses for a 64-tap design are shown in Figure 7 for each of the windowing methods mentioned above.

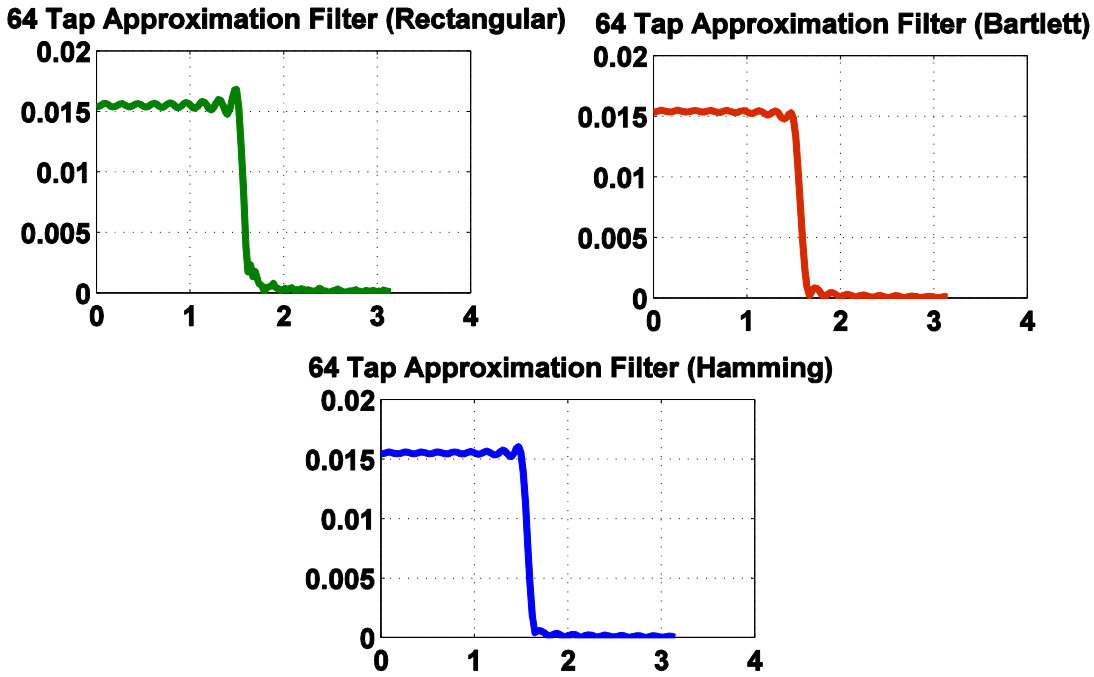


Figure 7: Approximation frequency responses for 64 tap filters using the proposed method.

A first revision of the approximation unit in hardware was created for feasibility analysis. The process was serialized using a state machine to traverse the design steps. The approximation circuit was implemented in Verilog and put through the first-step of the synthesis process, Cadence RC Compiler, to get an estimate for power and area. From the preliminary results using a 64-tap filter, the generated circuit would have an area of 62,422um^2 (2953 gates), an estimated dynamic energy of 20fJ, and a leakage energy of 0.75fJ at 0.5V and 200kHz. The SRAM active energy penalty was estimated to be 24.2fJ, with leakage energy of 13.18fJ, and the area penalty is ~17,613um^2. From these results, the approximation methods results in lower energy, but still falls short in area and throughput.

2.7.2. Test Chip Design

To test the feasibility of this method in silicon, the first iteration of a coefficient approximation generator was designed and fabricated in a 130nm technology with a block diagram shown in Figure 8. The design was completely synthesized from a Verilog description with the exception of the SRAM. To fairly evaluate the energy efficiency of the coefficient approximation design, a subthreshold SRAM, designed by Jim Boley, was included. The FIR core has a resource shared structure as described in section (2) and can receive coefficients from the approximation unit or the SRAM determined by the mux select bits. Scan chain bits are used to program the header switches for all blocks to evaluate leakage energy for both methods. Future iterations of the approximation unit will use a register file for energy and area comparisons, but it was not included in this revision.

2.8. Contributions

Contributions and metrics for success (sub-bullets) include:

2. A completed coefficient approximation methodology and circuit for a FIR filter
    a. A working design in silicon for (at minimum) FIR coefficient generation. If time permits, an FFT twiddle factor generator will also be included.
    b. Include register file in next iteration for a lookup table energy comparison

c. A CORDIC processor designed by Patricia Gonzalez was taped out on the last revision of the BSN node. CORDIC processors can compute trigonometric functions given an input angle, but generally have higher latency and power than the proposed method. The results of testing the CORDIC block will be included in the comparison for trigonometric function approximation.

3. Explore the accuracy-energy tradeoffs of using other, novel approximation methods. As an example, a five segment linear approximation for a sine is shown in Figure 9, and requires fewer multiplication operations compared to the quadratic approximation discussed in this section and used in the taped out design.
   a. In addition to presented approximation methods, evaluate segmented linear and direct sinc function approximations.

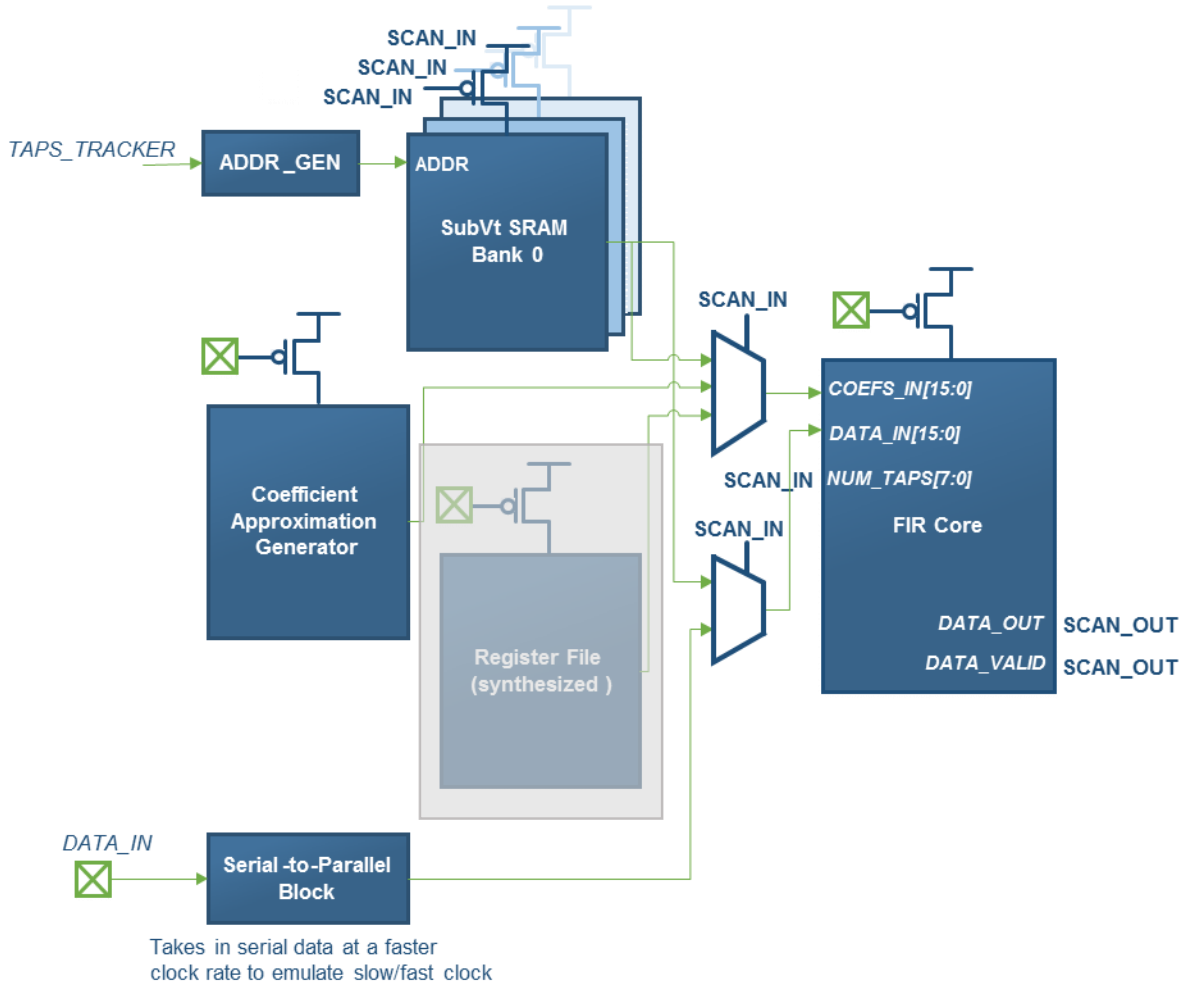
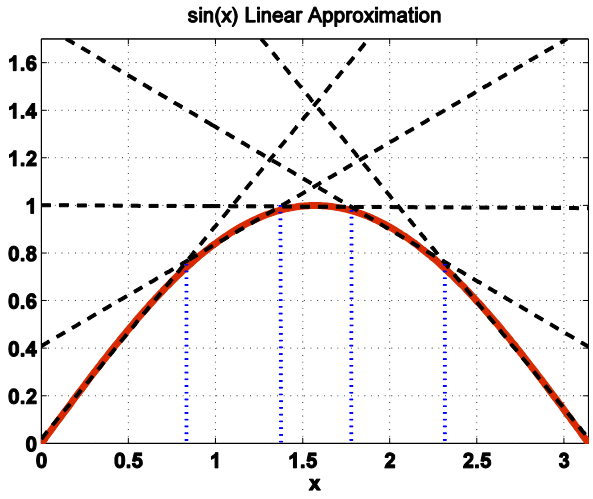


Figure 8: FIR Approximation test design including comparisons against SRAM and register files. This diagram includes information about input and output pads as well as scan chain programmability. The register file was not included during this tapeout due to area constraints on the test chip.

4. Completed analysis on the limitations of using this method for filtering. Filters with a small number of taps (< 16) will most likely not benefit from using this method, so analysis of the test circuit is required to determine the "tipping point" between the two methods.
5. Apply this methodology to IIR filters to create instability resilient adaptive IIR filters. In section (2) of this document, the instability of IIR filters was discussed in the BSN application space. If overflow (instability) is detected while filtering, dynamic coefficient generation could relax the specifications in real-time to prevent future errors. Although IIR coefficient design requires a different methodology (e.g. bilinear transforms), fewer coefficients need to be computed to achieve the same performance as an FIR.
    a. Do a theoretical (MATLAB) analysis of IIR coefficient methods to determine feasibility.



6.

Figure 9: Linear Approximation of sine using 5 segments. Method only requires one multiplication per estimation versus two for the quadratic approximation.

Expected challenges:

1. The use of coefficient approximation increases the latency to process one sample of data. Each coefficient can take 5-8 clock cycles to generate, and then additional cycles are required to process each sample. Depending on the required accuracy of the filter, this limits the scope of using this technique to applications with data rates on the low end of the BSN spectrum (e.g. ExG). Part of this work will specify a set of biomedical applications where this could be used.
2. The current version only supports low and highpass coefficients. To generate bandpass coefficients, the convolution of bandpass and lowpass impulse responses is required. This is a costly operation and could reduce the energy savings of using this method. Part of this work will explore methods for the efficient generation of bandpass coefficients.

3. SoC Ultra-Low PoweR (SUPR) Model

3.1. Overview and Motivation

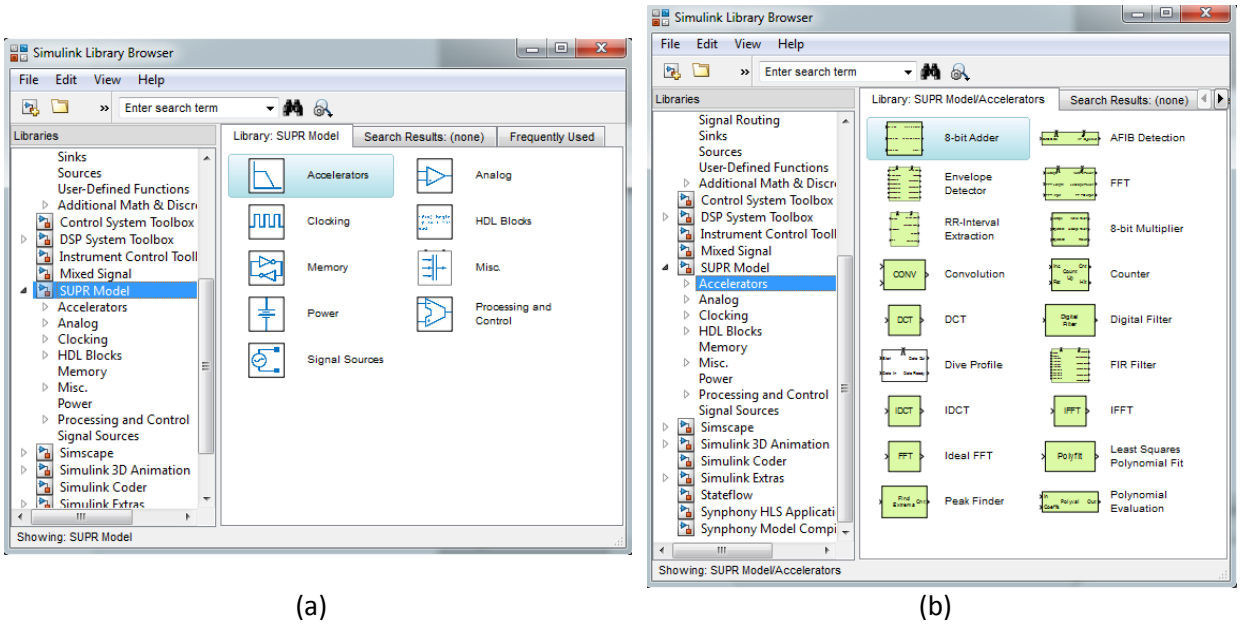
The integration density of modern ASICs is increasing and the functionality is becoming ever more complex. The developer has to design and optimize both hardware and software in order to build a fast and efficient system. Component and system modeling enables the designer to detect and overcome design bottlenecks in a fast and accurate manner. For ASIC designers, it's particularly difficult

to evaluate design tradeoffs when circuit modeling tools such as SPICE or Spectre can take days to run large simulations for even hundreds of microseconds. Not evaluating alternative designs or system architectures can lead to a poorly optimized system or component, and mistakes are costly when fabricating a custom chip. If circuit-level results show that the design does not meet specifications, the designer must return to the HDL to make modifications and go through another iteration of synthesis and simulation. A tool is needed that provides top-down system and block-level design space exploration in a fast and user-friendly way.

There are many available tools for understanding the performance and behavior of BSNs, with varying levels accuracy, scalability, and user feedback. Very few include low-level information on the underlying system hardware including energy and delay analysis. Current state-of-the art uses Synphony, which is a library of configurable hardware blocks that can be used within the Simulink environment and in a traditional synthesis flow [22][23]. Synphony blocks require expensive licenses and the user is limited to their provided block set making it unattractive for a scalable model. Other presented models have used SystemC, a C-like HDL, for BSN system and network modeling for a commercial off-the-shelf (COTS) BSN running TinyOS [24].

SUPR, the custom modeling tool discussed in this section, is implemented in Simulink, which is a MATLAB simulator that was chosen due to its prevalence and familiarity among students and academia. Simulink is industry standard for signal processing and provides discrete and continuous time signal-flow modeling. Similarly, the HDL simulation and verification tool that MATLAB uses for co-simulation is ModelSim, a simulation tool that our research group and many other HDL designers frequently use. Simulink's HDL Coder is a tool that can transform a Simulink model to an HDL description (VHDL or Verilog). Standard Simulink blocks and state machine diagrams can use HDL Coder to quickly generate a hardware description of a high-level model for comparison of key metrics.

Simulink models are well-suited for dedicated DSP algorithms such as filters (Figure 11) and FFTs as well as hierarchical macros such as ADPLLs. Common issues that need to be handled with simulation environments include precision, validation, scalability, and user-friendliness.



(a) (b)

Figure 10: (a) SUPR Model Library integrated into Simulink's library hierarchy (b) Example set of blocks provided in the accelerators library.

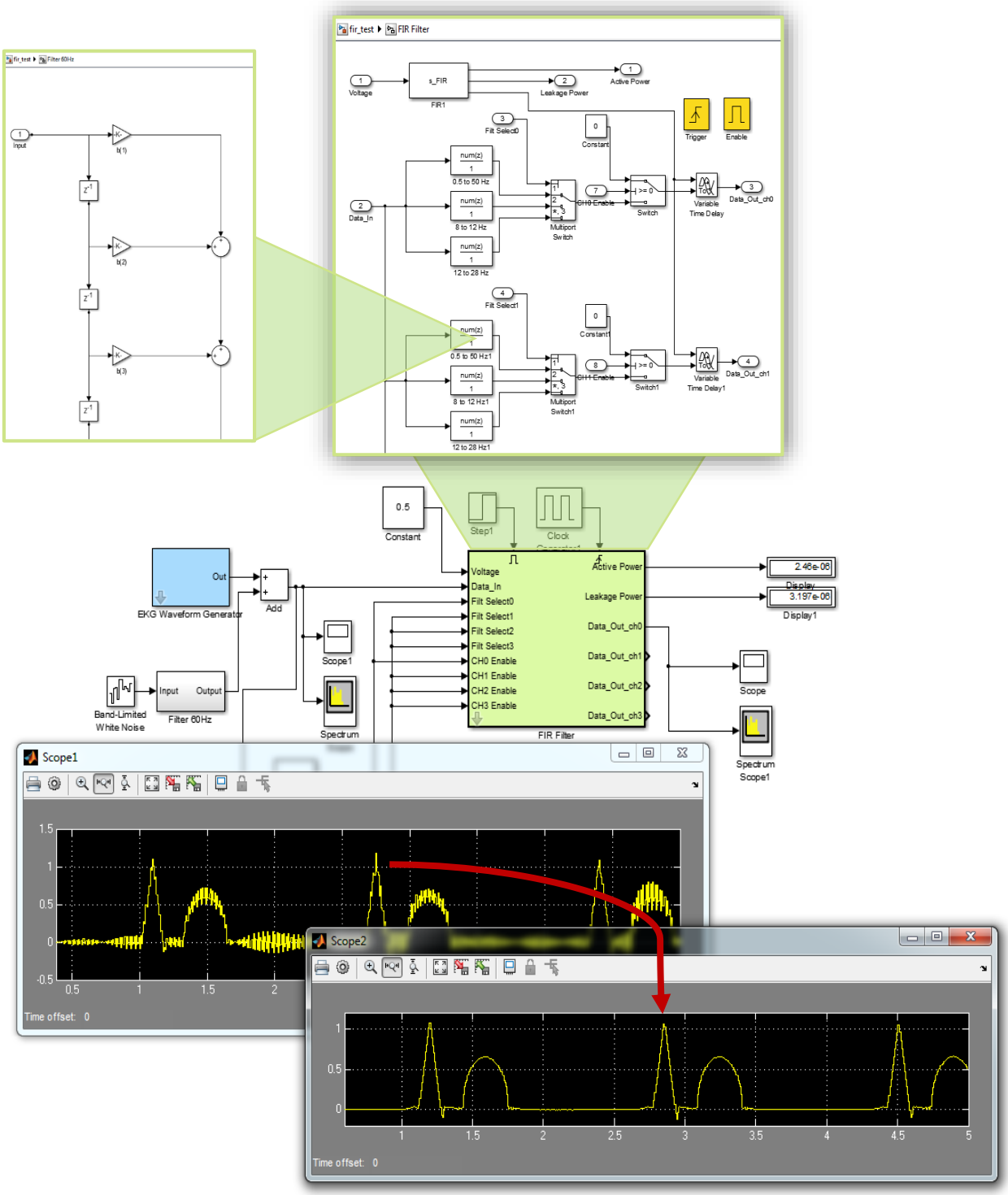


Figure 11: Example model for FIR filter removing power line interference (60Hz) noise from an EKG signal. This figure shows the hierarchy of design from behavioral descriptions down to functional.

1.1. Hypothesis

The proposed modeling tool and infrastructure will aid in justifying architecture-level design decisions for quick comparisons to design alternatives. Designs can be represented using standard Simulink blocks, MATLAB functions, and FSM structure and then translated into HDL code for synthesis. SUPR will form a full design loop between high-level algorithm description and physical design evaluation.

1.2. Approach

In order to have a modeling tool that's used to explore the design space for a low-power SoC, a library of common chip blocks needs to be completed. Many on-chip blocks have already been created in the current SUPR library including accelerators, clocking support, processing and control, analog, memory, and common biomedical signal sources taken from open-source biomedical databases. Each block in the model requires an input voltage and clock source to determine the energy and delay for that operating point. Delay information is sent to a variable time delay Simulink block at the output to delay the outputs and accurately represent real circuit performance. If the behavior of the circuit can be captured using Simulink library blocks or MATLAB code, a behavioral model is included.

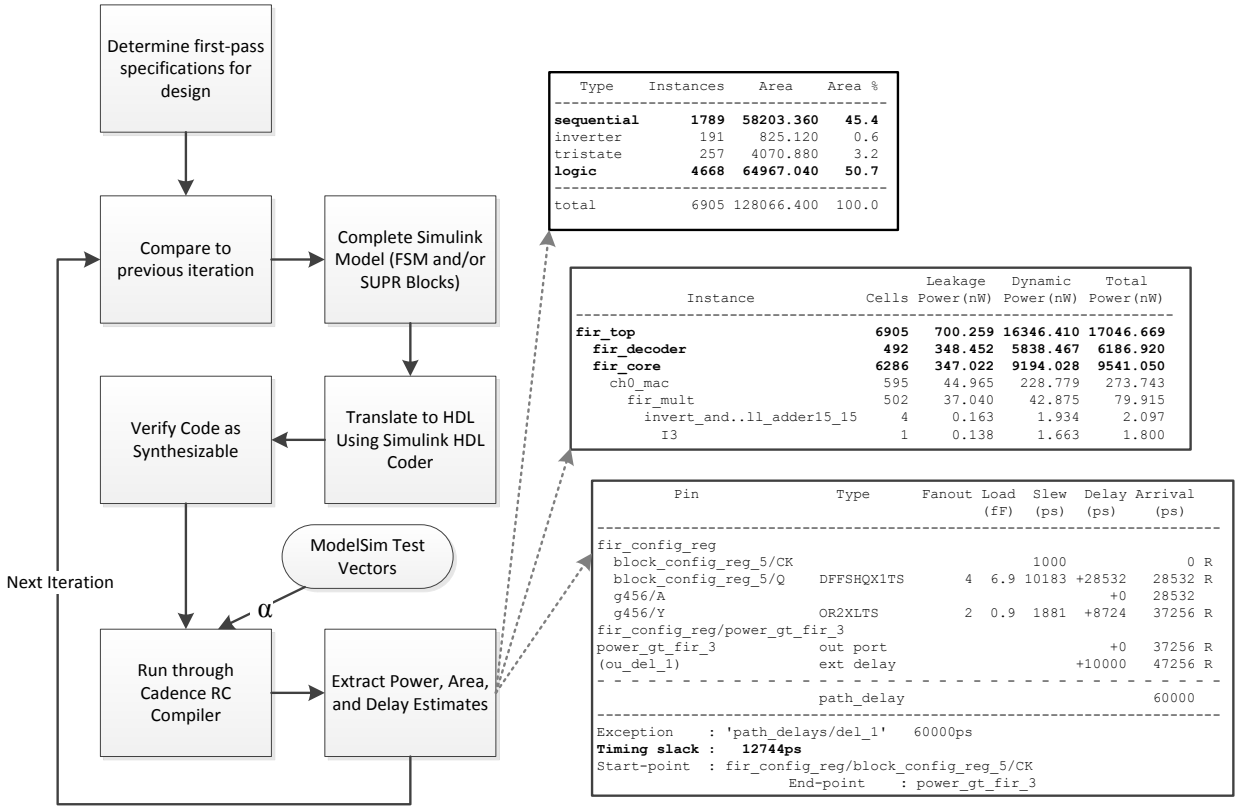


Figure 12: Simulink flow for hardware specification, design, and comparison. Example output files from Cadence RC Compiler are shown and parsed by MATLAB for comparison to previous designs.

Being a group that designs biomedical systems on chip, we often test accelerator blocks or algorithms using medical data from open-source databases such as PhysioBank [16]. Oftentimes these data are in inconvenient formats (e.g. European data format) and were sampled at rates that aren't being used in the modeled system. Data for EEG (motor movement, sleep) and EKG (normal and atrial defribrillation) have been imported into MATLAB formats and incorporated into a hierarchical block with interpolation and decimation options. Similarly, user-defined voltage or current profiles are needed to model the voltage on a storage capacitor over time due to energy harvesting, for example. A GUI has been developed to graphically define voltage/current profiles with custom sampling rates. The file can then be imported into a "custom signal" SUPR block.

To build a flexible tool that doesn't limit the hardware designer, a set of block primitives needs to be provided. Since many blocks in a SoC do not simply consist of well-known functions such as adders and multipliers, a method to integrate custom logic into the flow is necessary. Simulink's Stateflow toolbox allows for FSM-level descriptions including event-triggered systems, hierarchy, concurrency/parallelism, and variable support found in most programming languages. Stateflow models can be seamlessly mixed with Simulink models containing the standard block set. Using a dataflow graph or state machine to represent functionality is helpful not only to the designer but to future users/modifiers of the block. Reading and modifying HDL written by others (if not well-documented) can take time to be fully understood as HDL code can be written structurally (low-level gate description) to behaviorally (C-like description). The purpose of the final model will be to accurately capture the digital side of the chip while making it easy to scale the existing model for the addition of analog or mixed-signal blocks.

The approach for the final design space exploration tool is shown in Figure 12. The first step is to have an idea of the specifications or a metric that should be optimized and a black-box description of the block to be modeled. Realistic test vectors should be included for the design to extract circuit activity factor, α, used in synthesis for reliable power estimates. If it's the first iteration of the design, the Simulink model needs to be created using MATLAB functions, Simulink models, and Stateflow graphs. SUPR then sends the generated HDL through the Cadence's RC Compiler, a tool that generates a technology dependent gate-level netlist. RC Compiler also provides initial estimates of power, area, and delay that will be displayed in a GUI for the user to compare to previous designs in the SUPR database.

1.3. Proposed Contributions

Contributions and metrics for success (sub-bullets) include:

1. Complete a diverse library of system blocks that include energy and delay information for a given voltage-frequency point
    a. What will be considered "complete" for the dissertation is to have models for every block currently contained in the most recent revision of the BSN chip (Figure 2). Only the digital blocks will have low-level functional models, while mixed-signal and analog blocks will contain energy, delay, and high-level behavior descriptions.
2. GUI for power, energy, and delay feedback for a block or system model.
    a. Simple MATLAB generated GUI with a chart layout that has a backend to parse the output files of the synthesis step, extract needed information, and display it to the user.
3. Modeled replica of BSN node from [2] for system modeling validation. Example System-level questions the model should be able to address include:
    a. What sized memory (instruction and data memory) on chip will lead to the least amount of leakage while still accommodating the application?
    b. What's the minimal instruction set architecture needed for the LCU for low power and complete bus control?

c. An analysis of the model accuracy compared to measured results from the chip will be completed.

4. Thorough documentation of the model for future use
    a. A person that is unfamiliar with Simulink, but *is* familiar with hardware design could use the documentation to analyze design tradeoffs using this framework.

Expected challenges:

1. Seamless automation between Simulink and Cadence tools
2. Limited set of Simulink components can be used for HDL code generation
3. Model timescales determine the simulation runtime. This should be the smallest data rate in the system and sets the simulation time step. Multi-rate systems that have timescales that are orders of magnitude apart could take longer (minutes to hours) to simulate.
4. I know relatively little about the analog designs and radios on the BSN nodes. These models will either contain minimal information or data from state of the art literature
5. I want to avoid deterministic output data for energy and delay of the digital blocks. It would be best to represent these quantities as distributions with small variances. The main question would be determining the mean and variance values for each voltage-frequency point and seeing how the inclusion of nondeterministic data affects the runtime.
6. Capturing wire delay

2. Subthreshold All-Digital Phase-Locked Loop (ADPLL)

2.1. Overview and Motivation

For the SUPR modeling tool, it's important that the results are validated in silicon in order to show that the model can accurately predict circuit behavior. The plan for SUPR that was discussed in the previous chapter is to be a modeling tool for traditional DSP components and SoCs, but traditional DSPs don't capture the full range of complexities that could occur in an arbitrary digital block. Other digital designs can contain more complex data paths such as control logic, feedback, event-based operation, multi-rate operation, and multi-domain analysis (e.g. time, frequency, and phase). A digital circuit that contains many of these system elements is an all-digital phase-locked loop (ADPLL). An ADPLL circuit has applications in GHz rate digital communications, clock and data recovery, and for clock synthesis of an on-chip crystal oscillator [26]. The focus of this proposal will be for the latter. PLL circuits can be used to take a low-frequency input source and generate a higher frequency output clock with lower power compared to using an additional high-frequency crystal oscillator.

Typical ADPLL designs include four stages shown in Figure 13: the time-to-digital converter (TDC) that determines the phase error between the two signals, the loop filter to smooth out high-frequency phase-error data, the digitally-controlled oscillator (DCO) that generates the output clock, and a frequency divider to divide the output frequency to match the reference.

SoC PLLs need to be low-power enough to support always-on operation as clock sources are required in active and standby modes. Another design challenge associated with PLLs is the lock time. Whenever a change in phase is detected in the reference signal, the PLL takes multiple cycles to successfully relock the phases of the two signals. Due to constant state changes on chip that require varying clock frequencies, the PLL should have a lock time that is small relative to the time it takes to change states [29] to keep system power low during transitions to critical chip modes. Due to a high-accuracy requirement for most PLLs with output frequencies in the GHz range, subthreshold operation has not been investigated.

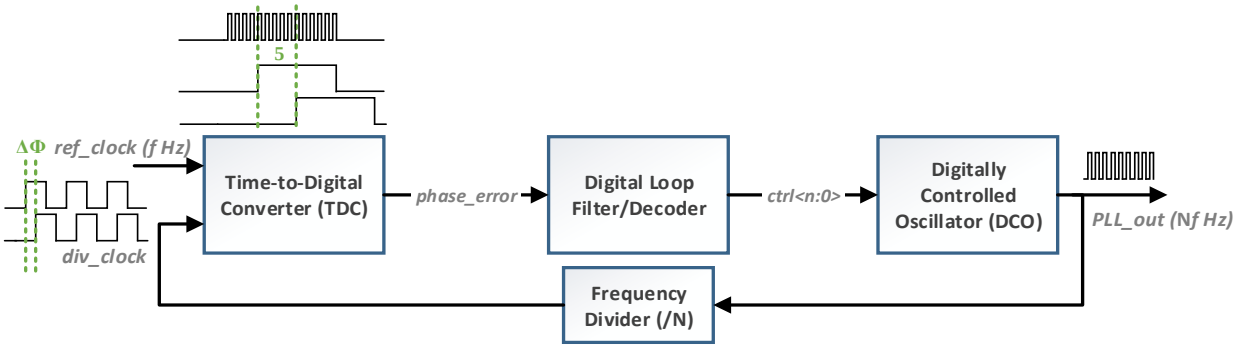


Figure 13: Block diagram of a typical ADPLL.

Crystal oscillators are used on-chip for low-jitter, accurate clock sources, but as the frequency of the crystal increases, the power tends to increase exponentially [32]. On flexible BSNs the overhead of adding extra crystal oscillators to supply multiple clock domains is not feasible due to power and the added area due to the circuit passives, but components such as the digital, analog and RF all require different clock frequencies. Unlike a traditional analog PLL, an ADPLL requires that all intermediate (control) signals are digital although many designs still contain analog components.

As integration and portability have become more important with CMOS process scaling, much emphasis has been placed on finding digital implementations for analog components in a PLL. For a traditional PLL, the use of many analog passives such as resistors and inductors with high quality factors requires intensive layout and need to be repeated in every technology node. It is desirable to have HDL descriptions of all PLL blocks so that the circuit can be easily replicated in any technology using standard cells for synthesis. One of the main goals for this project is to have a fully-synthesizable ADPLL, which is rarely attempted in the literature. Also, by using all digital components, the immunity to supply noise and temperature variation is improved and it will result in a more compact implementation as the circuit can take advantage of CMOS technology scaling [33].

Due to the various topologies, locking algorithms, and applications for an ADPLL, making design decisions and tradeoffs can be difficult due to the time involved comparing old methods and novel ideas [27]. The SUPR framework will have an ADPLL library for comparison of designs and backend analysis for plotting key metrics such as phase and quantization noise.

2.2. Approach

A subthreshold ADPLL was designed and fabricated in a commercial 130nm technology as a first iteration design to determine feasibility for a subthreshold ADPLLs. No modeling was completed in SUPR for this iteration, but the components will be modeled and synthesized using the proposed framework to be used as a base case ADPLL model. The components used in this design are the same as those described in Figure 13 and are explained in more detail in the following sections.

2.2.1. Time-to-Digital Converter (TDC)

The TDC is often the most power hungry block in a PLL design due to the requirement of a fast oscillator used to quantize the time difference between the phases [34]. Three types of TDCs are commonly used in designs: the delay-line based TDC, the ring oscillator based TDC, and the two-step TDC [28]. Two topologies were investigated for the TDC used in this ADPLL. The first, a Vernier delay technique (delay-line), determines the reduced delay between the two signals by taking a difference of

the delays. For this approach a large amount of area and delay stages would be required to encompass the entire reference clock period (microseconds). The second design was an oscillator-based approach that used a fast oscillator to count the number of pulses that occurred during the phase error window. Using this approach, a large range can be achieved with compact area and the quantization error becomes scrambled across measurements. The proposed approach is not often seen in ADPLLs since the Vernier delay approach can be used for high-frequency applications with little area penalty [26].

The completed TDC used a ring-oscillator based design An 8-bit counter was used so that 157 clock pulses were needed to cover a full period of the reference clock using a TDC fast clock with a 200ns period to achieve the desired resolution. The number of pulses in the phase error is counted and passed to the "DCO Controller" stage. The TDC logic runs at 500mV and consumes 240nW when operating with a reference frequency of 32kHz from simulation results.

State of the art TDC designs have been introduced that attempt to reduce the power of the fast oscillator without hurting the performance. One such TDC uses an enable signal on the delay line of the fast oscillator to disable inverter switching when the phase error is not being evaluated [37]. Other designs have used AND gate based ring-oscillators where one input of the AND is used as a delay stage clock gate [38].

2.2.2. DCO Controller

The counter word received from the TDC must be decoded into a set of bits that determines the number of active delay stages in the ring oscillator. To set the output frequency. Many DCO controllers use a binary search algorithm to determine the locking frequency as in [30][35] or have phase error prediction logic as in [34]. This implementation used a binary search that is initialized in the center of the delay cells for shorter locking time. In order to increase the accuracy and decrease the locking time, frequency acquisition is completed in two steps using coarse and a fine delay lines as shown in Figure 14.

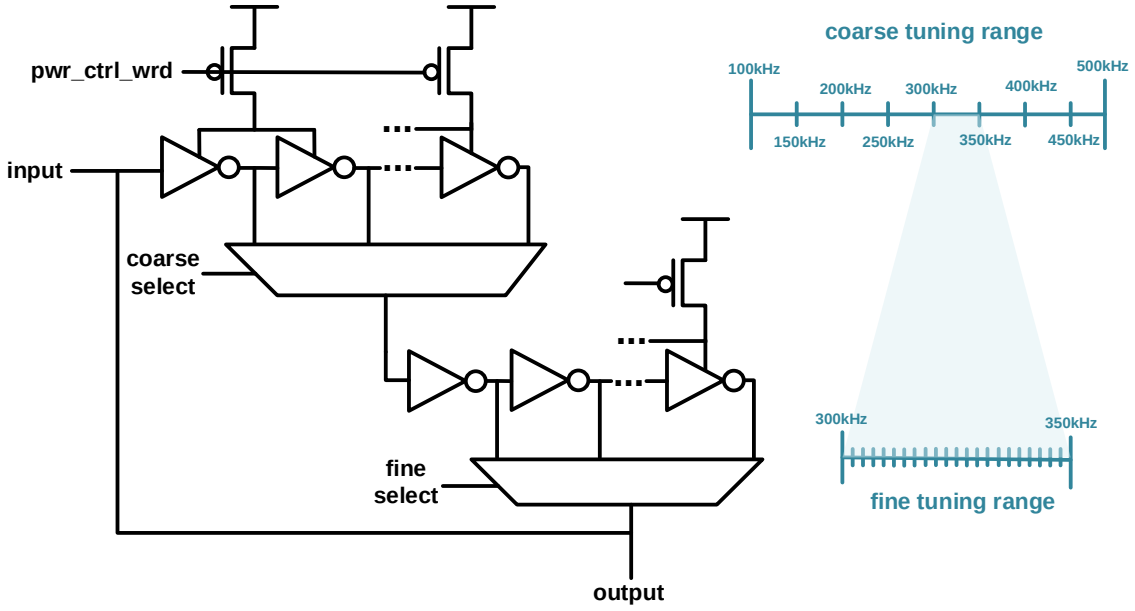


Figure 14: DCO block diagram including the fine and coarse stages for the ring oscillator.

The coarse stage is used to narrow the output frequency search range, while the fine tuning stage searches within the coarse range using delay cells with a smaller delay (leading to a higher resolution). The output of the DCO controller is a series of control bits for the select lines of the muxes. Once the ADPLL has locked, there are controller output bits to control the PMOS headers of the delay cells in the DCO shown in Figure 14. If any delay cells are not being used, they are power gated to reduce the switching power in the delay line. The total simulated power consumption for the control logic and the frequency divider circuit was 6nW at 500mV.

2.2.3. Digitally Controlled Oscillator (DCO)

In many ADPLL designs, the DCO can consume 50-70% of the overall power budget [31]. There are various topologies that are often used for ring-oscillator design. The most common delay elements used for low-power operation are shown in Figure 15. Before the DCO was designed, different delay cell topologies were surveyed from the literature and simulated to determine their delay to power performance. Four delay cells were encountered most frequently in ring-oscillators and are the inverter, the shunt capacitor inverter, the hysteresis delay cell (HDC), and the current-starved inverted. Many DCOs in the literature also focus on high performance RF applications and used LC tank oscillators due to their low phase noise and high performance compared characteristics [36]. LC tank oscillators are not portable due to the custom layout of the capacitor and inductor, and at lower frequencies the passive components for become large and are no longer feasible [35].

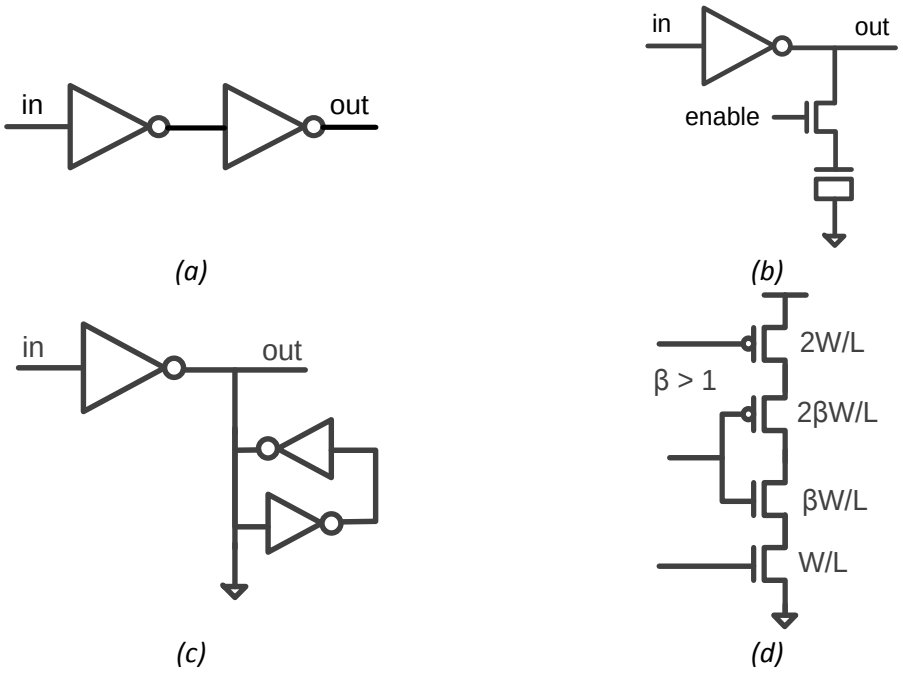


Figure 15: Common delay cell topologies for the DCO (a) Inverter (b) Shunt capacitor (c) hysteresis delay (d) current-starved.

To determine which delay cell to use, a sweep of the output frequency and average power consumed per period was found for each of the delay cell topologies. The HDC can achieve small delays, but at the penalty of a high power consumption. HDCs also rely on ratioed logic that tends to not be reliable in the subthreshold regime due to sensitivity to variations. The inverter achieved very small delays for a small amount of power, and was used for the fine tuning stage. Although the current-starved topology can

also achieve very small delays, it consumed more power and required a current reference, which is not possible for synthesized designs. The shunt capacitor inverter was used for the coarse stage due to its relatively high delay and low power consumption. The total simulated power for the DCO running at 512kHz was 81nW at 500mV.

2.2.4. ADPLL Simulation Results

The completed PLL uses a low frequency reference of 32kHz and has an output frequency in the range of 200kHz-1MHz using an integer-N ADPLL approach (the output frequency can only be integer multiples of the input frequency). The design consumes less than 400nW of power while running all circuits in the subthreshold region and reduces the locking time to a maximum of 8 clock cycles by using a ring-oscillator based time-to-digital converter. The ADPLLs simulated locking behavior is shown in Figure 16.

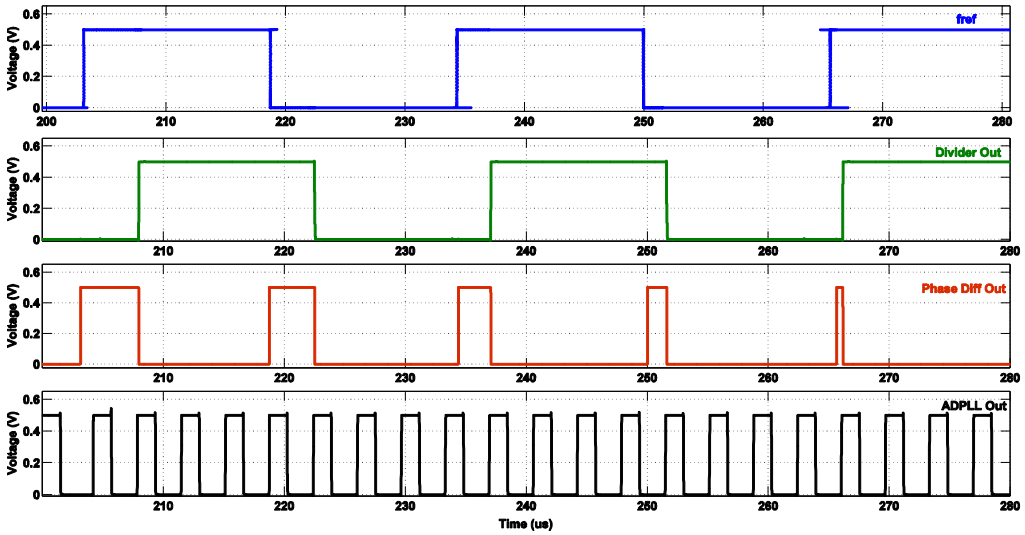


Figure 16: ADPLL locking to the reference signal.

The SUPR environment allows for multi-domain analysis of ADPLLs so that time and phase-domain models can be used to converge to the final design. Phase domain simulations are used for the analysis of noise (e.g. phase and quantization), while time domain models provide information regarding functionality such as lock time as seen in Figure 17 in an exampletime-domain model for a PLL. Important metrics to report for ADPLLs include reference frequency, linear frequency range, frequency spacing, lock time, phase noise (dBc/Hz), and power consumption. Generated models will report these metrics in either tabular or graphical form for comparison. For the next iteration of the subthreshold ADPLL design, the SUPR environment will be used to model a fractional-N ADPLL for on-chip clock synthesis using a 50kHz reference and converting up to 187.5kHz, the clock used for digital operation.

2.2.5. Contributions and Anticipated Challenges

Contributions and metrics for success (sub-bullets) include:

5. A completed ADPLL model for the circuit described in this section.
    a. Completely synthesizable (exception: DCO) through the SUPR framework

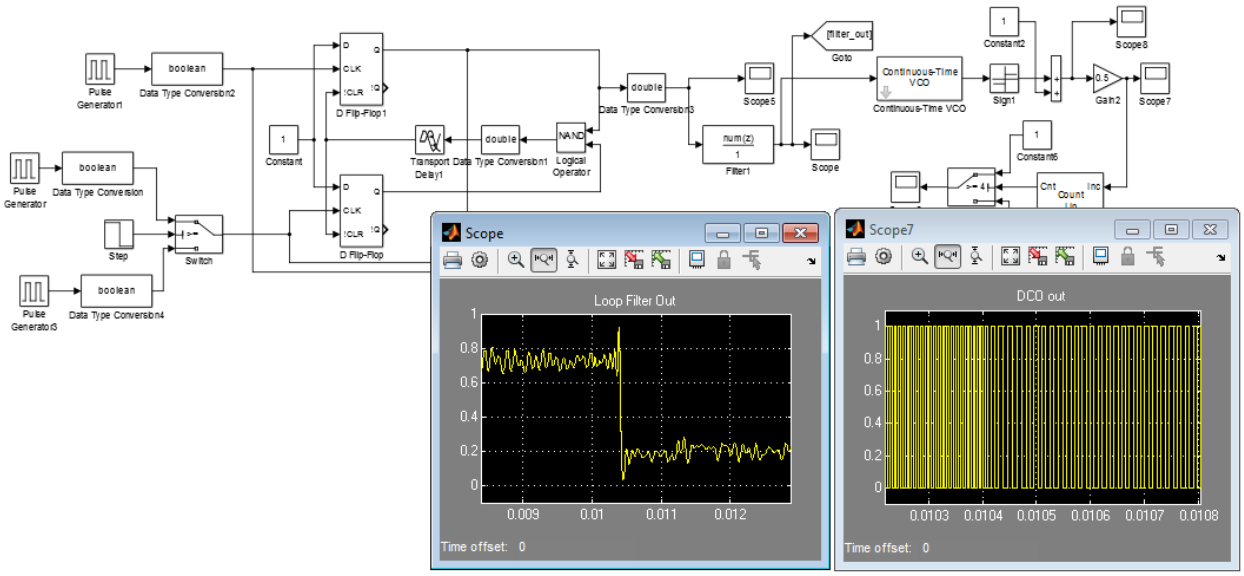


Figure 17: Model of feed-forward path for ADPLL in SUPR showing the output of the loop filter changing with a corresponding frequency shift in the DCO.

1. Investigate loop filter topologies (FIR, IIR) in SUPR to reduce the overall jitter and lock time of the ADPLL.
   a. Compare performance (locking time), area, and power of two loop filter architectures (IIR, FIR) with varying amounts of taps.
2. Explore locking algorithms and how they affect power of the overall APLL
   a. Analyze binary search, phase prediction, and brute force algorithms to replace the loop filter.
3. Add multiple types of delay cell topologies to the model (the four in Figure 15)
   b. Information regarding delay with respect to supply voltage, power with respect to supply voltage, and phase noise of each cell. Integrate this information with a behavioral Simulink oscillator.
4. Design and fabricate a subthreshold integer-N ADPLL.
5. Design and fabricate a fractional-N ADPLL using results from SUPR.
6. Thorough documentation of the ADPLL model for future use.

Expected challenges:

1. Accurate modeling of ring oscillator circuits (synthesizing loops will not work using EDA tools)
2. Determining whether design errors are due to design choices or modeling error (e.g. for a poor phase noise result, was the DCO not represented correctly in the model or is the phase noise actually bad for this DCO design?)

3. Acknowledgments

The BSN chips that were discussed in this document had many other contributors besides myself. The University of Washington designed the analog front-end (AFE) for the first revision of the chip and The University of Michigan designed the radios (transceiver and wake-up radio) for the second revision of the chip. I'd also like to acknowledge my lab mates that also worked on either revision of the chip: Yousef Shakhsheer, Yanqing Zhang, Kyle Craig, Jim Boley, Aatmesh Shrivastiva, Patricia Gonzalez, Divya Akella, and my advisor, Ben Calhoun. For the modeling effort, I had the opportunity to mentor four undergraduates that generated sub-libraries and ideas for the future of the model: Chuhong Duan, Roman Boyarov, Ian Dansey, and Vincent Luu.

4. Research Tasks and Schedule

Subject	#	Task Description	Status	Related Publications
(BSN v0) Subthreshold FIR Filter	1	Architecture Exploration	Completed	
	2	HDL Description/Simulation	Completed	
	3	Synthesis, Place and Route, Verification	Completed	
	4	Chip Testing	Completed	[AK 0][AK 1][AK 2]
BSN v1/v2	1	Chip Architecture Exploration	Completed	
	2	Block Level Architecture Explorations (FIR, FFT, Dive Profile, I2C/SPI, etc.)	Completed	
	3	HDL Description/Simulation	Completed	
	4	Synthesis, Place and Route, Verification	Completed	
	5	BSN v1 Chip Testing	Sep-13	[AK 3] [AK 4]
	6	BSN v2 Modifications	Aug-13	
	7	BSN v2 Block Synthesis/Layout	Aug-13	
	8	BSN v2 Chip Testing	Jan-14	[AK 5]
FIR Approximation	1	Architecture Choice	Completed	
	2	HDL Description/Simulation	Completed	
	3	Synthesis	Completed	
	4	Silicon Testing	Sep-13	[AK 6]
	5	Revision 2 HDL Description	May-13/Aug-13	
	6	Synthesis	May-13/Aug-13	
	7	Chip Testing	Jan-14	[AK 7]
SUPR	1	Framework Setup	Completed	
	2	Core Block Design	Completed	
	3	Library of Chip Blocks to Match BSN	Dec-13	
	4	Stateflow Integration	Jan-14	[AK 8]
	5	Statistical Integration of Energy/Delay	Feb-14	
	6	User Interface	Aug-14	
	7	Simulink-to-HDL	Jun-14	[AK 9]

ADPLL	1	Version 0 (v0) Architecture Choice	Completed	
	2	v0 HDL Description/Simulation	Completed	
	3	v0 Synthesis	Completed	
	4	v0 Chip Testing	Jun-13	[AK 10]
	5	SUPR ADPLL Framework	Nov-13	[AK 11]
	6	ADPLL v1 Design and Synthesis	May-14	
	7	ADPLL v1 Chip Testing	Dec-14	[AK 12]
Thesis Writing	1	Write final publications, await decisions	Jan-15	
	2	Write Thesis	Apr-15	
	3	PhD defense	May-15	

5. Publications

5.1. Completed

[AK 0] **Klinefelter, A.;** Yanqing Zhang; Otis, B.; Calhoun, B.H., "A Programmable 34 nW/Channel Sub-Threshold Signal Band Power Extractor on a Body Sensor Node SoC," Circuits and Systems II: Express Briefs, IEEE Transactions on , vol.59, no.12, pp.937,941, Dec. 2012.

[AK 1] Yanqing Zhang; Fan Zhang; Shakhsheer, Y.; Silver, J.D.; **Klinefelter, A.;** Nagaraju, M.; Boley, J.; Pandey, J.; Shrivastava, A.; Carlson, E.J.; Wood, A.; Calhoun, B.H.; Otis, B.P., "A Batteryless 19µW MICS/ISM-Band Energy Harvesting Body Sensor Node SoC for ExG Applications," *Solid-State Circuits, IEEE Journal of* , vol.48, no.1, pp.199,213, Jan. 2013.

[AK 2] Fan Zhang; Yanqing Zhang; Silver, J.; Shakhsheer, Y.; Nagaraju, M.; **Klinefelter, A.;** Pandey, J.; Boley, J.; Carlson, E.; Shrivastava, A.; Otis, B.; Calhoun, B., "A batteryless 19µW MICS/ISM-band energy harvesting body area sensor node SoC," *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International* , vol., no., pp.298,300, 19-23 Feb. 2012.

5.2. Planned

[AK 3] Next revision BSN paper on the entire working chip
[AK 4] Paper on low-power accelerators and novel DSP on the BSN chip
[AK 5] BSN revision v2: summary of the entire chip
[AK 6] FIR approximation unit for sub-threshold operation in SoCs
[AK 7] Approximations for classical DSPs (FIR, IIR, FFT) on low-power DSPs
[AK 8] A model for BSN design space exploration using Simulink
[AK 9] An automated and high-level design flow for digital circuit design using Simulink
[AK 10] A Subthreshold ADPLL for low-rate Clock Synthesis on SoCs
[AK 11] A model for Exploring Tradeoffs of ULP ADPLLs using Simulink
[AK 12] A Subthreshold ADPLL for ULP Clock Synthesis with SUPR Model Support

6. References

[1] I. Korhonen, J. Parkka, and M. Van Gils, “Health monitoring in the home of the future,” IEEE Engineering in Medicine and Biology Magazine, vol. 22, pp. 66–73, May-Jun. 2003.

[2] Zhang, F., et al., " A Batteryless 19µW MICS/ISM-Band Energy Harvesting Body Area Sensor Node SoC," International Solid-State Circuits Conference (ISSCC), 2012 IEEE, Feb. 2012.

[3] Zhang, Y., Y. Shakhsheer, A. T. Barth, H. P. C. Jr., S. A. Ridenour, M. A. Hanson, J. Lach, and B. H. Calhoun, "Energy Efficient Design for Body Sensor Nodes", Journal of Low Power Electronics and Applications, 04/2011.
[4] Davis, W.R. , et al., "A design environment for high throughput, low power dedicated signal processing systems," Custom Integrated Circuits, 2001, IEEE Conference on, 2001.
[5] Ming-Hung Chang; Yi-Te Chiu; Shu-Lin Lai; Wei Hwang; , "A 1kb 9T subthreshold SRAM with bit-interleaving scheme in 65nm CMOS," Low Power Electronics and Design (ISLPED) 2011 International Symposium on , vol., no., pp.291-296, 1-3 Aug. 2011.Pfurtscheller, G.; Neuper, C.; Guger, C.; Harkam, W.; Ramoser, H.; Schlogl, A.; Obermaier, B.; Pregenzer, M., "Current trends in Graz brain-computer interface (BCI) research," Rehabilitation Engineering, IEEE Transactions on, vol.8, no.2, pp.216-219, Jun 2000.
[6] Girard, Olivier. “openMSP430”. http://opencores.org/project,openmsp430
[7] Oppenheim, A., & Schafer, R. (2010). Discrete-time signal processing. (3rd ed.). Upper Saddle River, NJ: Pearson.
[8] Powell, S.R.; Chau, P.M., "A technique for realizing linear phase IIR filters," Signal Processing, IEEE Transactions on , vol.39, no.11, pp.2425,2435, Nov 1991.
[9] Avestruz, A.-T.; Santa, W.; Carlson, D.; Jensen, R.; Stanslaski, S.; Helfenstine, A.; Denison, T., "A 5µW/Channel Spectral Analysis IC for Chronic Bidirectional Brain–Machine Interfaces," Solid-State Circuits, IEEE Journal of, vol.43, no.12, pp.3006-3024, Dec. 2008.
[10] Zhang, F., et al., "A low-power multi-band ECoG/EEG interface IC," Custom Integrated Circuits Conference (CICC), 2010 IEEE, Sept. 2010.
[11] Abdelhalim, K.; Genov, R., "915-MHz wireless 64-channel neural recording SoC with programmable mixed-signal FIR filters," ESSCIRC (ESSCIRC), 2011 Proceedings of the, vol., no., pp.223-226, 12-16 Sept. 2011.
[12] Verma, N.; Shoeb, A.; Bohorquez, J.; Dawson, J.; Guttag, J.; Chandrakasan, A.P.; , "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System," Solid-State Circuits, IEEE Journal of , vol.45, no.4, pp.804-816, April 2010.
[13] Myeong-Eun H., et al., “A 85mV 40nW Process-Tolerant Subthreshold 8x8 FIR Filter in 130nm Technology," VLSI Circuits, 2007 IEEE Symposium on, June 2007.
[14] Wei-Hsiang M., et al., "187 MHz Subthreshold-Supply Charge-Recovery FIR," Solid-State Circuits, IEEE Journal of, April 2010.
[15] Calhoun, B.H.; Ryan, J.F.; Khanna, S.; Putic, M.; Lach, J. , "Flexible Circuits and Architectures for Ultralow Power," *Proceedings of the IEEE*, vol.98, no.2, pp.267-282, Feb. 2010.
[16] Brain Computer Interface research at NUST Pakistan. EEG motor Brain Computer Interface research at NUST Pakistan. EEG motor activity data set <https://sites.google.com/site/projectbci/>.
[17] Xin Xiao; Oruklu, E.; Saniie, J., "An Efficient FFT Engine With Reduced Addressing Logic," Circuits and Systems II: Express Briefs, IEEE Transactions on , vol.55, no.11, pp.1149,1153, Nov. 2008.
[18] Ming-Hung Chang; Yi-Te Chiu; Shu-Lin Lai; Wei Hwang; , "A 1kb 9T subthreshold SRAM with bit interleaving scheme in 65nm CMOS," Low Power Electronics and Design (ISLPED) 2011 International Symposium on , vol., no., pp.291-296, 1-3 Aug. 2011.
[19] Punskaya, E. (n.d.). Design of fir filters. Retrieved from http://wwwsigproc.eng.cam.ac.uk/~op205/3F3_5_Design_of_FIR_Filters.pdf.
[20] Baczynski, M. (2007, July 18). Fast and accurate sine/cosine approximation. Retrieved from http://lab.polygonal.de/?p=205.
[21] Green, R. (n.d.). Faster math functions. In Retrieved from http://www.research.scea.com/research/pdfs/RGREENfastermath_GDC02.pdf.
[22] Nanda, R.; Chia-Hsiang Yang; Markovic, D., "DSP architecture optimization in Matlab/Simulink environment," *VLSI Circuits, 2008 IEEE Symposium on*, vol., no., pp.192, 193, 18-20 June 2008.

[23] Nanda, R. “DSP Architecture Optimization in MATLAB/Simulink Environment” Master of Science in Electrical Engineering: Thesis. University of California, Los Angeles, 2008.
[24] Cutcutache, I.; Dang, T.T.N.; Leong, B.; Shanshan Liu; Nguyen, K.D.; Phan, L.T.X.; Sim, E.; Zhenxin Sun; Tok, T.B.; Lin Xu; Tay, F. E H; Weng-Fai Wong, "BSN Simulator: Optimizing Application Using System Level Simulation," Wearable and Implantable Body Sensor Networks, 2009. BSN 2009. Sixth International Workshop on , vol., no., pp.9,14, 3-5 June 2009.
[25] Mauderer, A.; Freier, M.; Oetjens, J.; Rosenstiel, W., "Efficient digital design for automotive mixed-signal ASICs using simulink," Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2012 IEEE 15th International Symposium on , vol., no., pp.372,377, 18-20 April 2012.
[26] Helal, B.M.; Straayer, M.Z.; Gu-Yeon Wei; Perrott, M.H.; , "A Highly Digital MDLL-Based Clock Multiplier That Leverages a Self-Scrambling Time-to-Digital Converter to Achieve Subpicosecond Jitter Performance," *Solid-State Circuits, IEEE Journal of* , vol.43, no.4, pp.855-863, April 2008.
[27] Mohn, R. “Modeling and Simulating an All-Digital Phase Locked Loop” Article Published by MathWorks, 2011.
[28] Abramovitch, Daniel, "Phase-locked loops: a control centric tutorial," American Control Conference, 2002. Proceedings of the 2002, vol.1, no., pp.1,15 vol.1, 2002.
[29] August, N.; Hyung-jin Lee; Vandepas, M.; Parker, R., "A TDC-less ADPLL with 200-to-3200MHz range and 3mW power dissipation for mobile SoC clocking in 22nm CMOS," Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International , vol., no., pp.246,248, 19-23 Feb. 2012.
[30] Doo-Chan Lee; Kyu-Young Kim; Young-Jae Min; Kyung-Min Kim; Abdullah, A.; Jongsun Park; Soo-Won Kim; , "A low power all-digital PLL with power optimized digitally controlled oscillator," Electron Devices and Solid-State Circuits (EDSSC), 2010 IEEE International Conference of , vol., no., pp.1-4, 15-17 Dec. 2010.
[31] M. Maymandi-Nejad and M. Sachdev, “A monotonic digitally controlled delay element,” IEEE J. Solid-State Circuits, vol. 40, no. 11, pp. 2212–2219, Nov. 2005.
[32] Vittoz, E.A., Degrauwe, M.G.R. and Bitz, S. High-performance crystal oscillator circuits: theory and application. IEEE journal of Solid-State Circuits, 23, 5 (Jun 1988), 774-783.
[33] Abadian, A.; Lotfizad, M.; Erfani Majd, N.; Ghoushchi, M.B.G.; Mirzaie, H.; , "A new low-power and low-complexity all digital PLL (ADPLL) in 180nm and 32nm," Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on , vol., no., pp.305-310, 12-15 Dec. 2010.
[34] Jingcheng Zhuang; Staszewski, R.B., "A low-power all-digital PLL architecture based on phase prediction," Electronics, Circuits and Systems (ICECS), 2012 19th IEEE International Conference on , vol., no., pp.797,800, 9-12 Dec. 2012.
[35] Tzu-Chi Huang; Hong-Yi Huang; Jen-Chieh Liu; Kuo-Hsing Cheng; Ching-Hsing Luo, "All digital phase-locked loop using active inductor oscillator and novel locking algorithm," Circuits and Systems (ISCAS), 2011 IEEE International Symposium on , vol., no., pp.486,489, 15-18 May 2011.
[36] Duo Sheng; Ching-Che Chung; Chen-Yi Lee , "An Ultra-Low-Power and Portable Digitally Controlled Oscillator for SoC Applications," Circuits and Systems II: Express Briefs, IEEE Transactions on , vol.54, no.11, pp.954-958, Nov. 2007.
[37] Jeong, C.-H.; Kwon, C.-K.; Kim, H.; Hwang, I.-C.; Kim, S.-W., "Low-power, wide-range time-to-digital converter for all digital phase-locked loops," Electronics Letters , vol.49, no.2, pp.96,97, January 17 2013.
[38] Olsson, T.; Nilsson, P., "A digitally controlled PLL for digital SOCs," Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on , vol.5, no., pp.V-437,V-440 vol.5, 25-28 May 2003.